AFRL-AFOSR-VA-TR-2016-0230

Architecture and Programming Models for High Performance Intensive Computation

XiaoMing Li
UNIVERSITY OF DELAWARE

06/29/2016
Final Report

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/ RTA2
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
06-27-2016	Final Report	09/30/2015-03/29/2016

4. TITLE AND SUBTITLE
Architecture and Programming Models for High Performance Intensive Computation

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA9550-13-1-0213

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Xiaoming Li, Guang Gao, Lian-Ping Wang and Jack B. Dennis

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Delaware
Office of Vice Provost for Research
Newark, Delaware 19716

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Office of Scientific Research (AFOSR)
875 N. Randolph Str.
Arlington VA 22203

10. SPONSOR/MONITOR'S ACRONYM(S)
AFOSR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This is the final report of our project. In this report, we will cover our research activities in the last report period, as well as summarize our achievements for the whole project. Our research effort has been directed at developing an efficient system architecture and software tools for building and running Dynamic Data Driven Application Systems (DDDAS). The foremost requirement for efficient operation of DDDAS is flexibility and efficiency of managing memory and processing resources for an operating environment characterized by continuously changing resource demands. The Fresh Breeze programming model is well matched to this application environment.
With generous support from AFOSR, and additional support from NSF, we have accomplished the an energy efficient computing platform that directly implements a programming model well suited to DDDAS. The details of our research activities and accomplishments are described in the separate project report.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE		14	Xiaoming Li
					19b. TELEPHONE NUMBER *(Include area code)* 302-831-0334

Reset

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18
Adobe Professional 7.0

INSTRUCTIONS FOR COMPLETING SF 298

1. REPORT DATE. Full publication date, including day, month, if available. Must cite at least the year and be Year 2000 compliant, e.g. 30-06-1998; xx-06-1998; xx-xx-1998.

2. REPORT TYPE. State the type of report, such as final, technical, interim, memorandum, master's thesis, progress, quarterly, research, special, group study, etc.

3. DATES COVERED. Indicate the time during which the work was performed and the report was written, e.g., Jun 1997 - Jun 1998; 1-10 Jun 1996; May - Nov 1998; Nov 1998.

4. TITLE. Enter title and subtitle with volume number and part number, if applicable. On classified documents, enter the title classification in parentheses.

5a. CONTRACT NUMBER. Enter all contract numbers as they appear in the report, e.g. F33615-86-C-5169.

5b. GRANT NUMBER. Enter all grant numbers as they appear in the report, e.g. AFOSR-82-1234.

5c. PROGRAM ELEMENT NUMBER. Enter all program element numbers as they appear in the report, e.g. 61101A.

5d. PROJECT NUMBER. Enter all project numbers as they appear in the report, e.g. 1F665702D1257; ILIR.

5e. TASK NUMBER. Enter all task numbers as they appear in the report, e.g. 05; RF0330201; T4112.

5f. WORK UNIT NUMBER. Enter all work unit numbers as they appear in the report, e.g. 001; AFAPL30480105.

6. AUTHOR(S). Enter name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. The form of entry is the last name, first name, middle initial, and additional qualifiers separated by commas, e.g. Smith, Richard, J, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES). Self-explanatory.

8. PERFORMING ORGANIZATION REPORT NUMBER. Enter all unique alphanumeric report numbers assigned by the performing organization, e.g. BRL-1234; AFWL-TR-85-4017-Vol-21-PT-2.

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES). Enter the name and address of the organization(s) financially responsible for and monitoring the work.

10. SPONSOR/MONITOR'S ACRONYM(S). Enter, if available, e.g. BRL, ARDEC, NADC.

11. SPONSOR/MONITOR'S REPORT NUMBER(S). Enter report number as assigned by the sponsoring/ monitoring agency, if available, e.g. BRL-TR-829; -215.

12. DISTRIBUTION/AVAILABILITY STATEMENT. Use agency-mandated availability statements to indicate the public availability or distribution limitations of the report. If additional limitations/ restrictions or special markings are indicated, follow agency authorization procedures, e.g. RD/FRD, PROPIN, ITAR, etc. Include copyright information.

13. SUPPLEMENTARY NOTES. Enter information not included elsewhere such as: prepared in cooperation with; translation of; report supersedes; old edition number, etc.

14. ABSTRACT. A brief (approximately 200 words) factual summary of the most significant information.

15. SUBJECT TERMS. Key words or phrases identifying major concepts in the report.

16. SECURITY CLASSIFICATION. Enter security classification in accordance with security classification regulations, e.g. U, C, S, etc. If this form contains classified information, stamp classification level on the top and bottom of this page.

17. LIMITATION OF ABSTRACT. This block must be completed to assign a distribution limitation to the abstract. Enter UU (Unclassified Unlimited) or SAR (Same as Report). An entry in this block is necessary if the abstract is to be limited.

Architecture and Programming Models for High Performance Intensive Computation

Xiaoming Li and Guang Gao and Lian-Ping Wang
University of Delaware
Jack B. Dennis
MIT Computer Science and Artificial Intelligence Laboratory

June 27, 2016

Summary

This is the final report of our project. In this report, we will cover our research activities in the last report period, as well as summarize our achievements for the whole project.

Our research effort has been directed at developing an efficient system architecture and software tools for building and running Dynamic Data Driven Application Systems (DDDAS). The foremost requirement for efficient operation of DDDAS is flexibility and efficiency of managing memory and processing resources for an operating environment characterized by continuously changing resource demands. The Fresh Breeze programming model [1][2] is well matched to this application environment.

With generous support from AFOSR, and additional support from NSF, we have accomplished the following steps toward an energy efficient computing platform that directly implements a programming model well suited to DDDAS.

- Create the simulation tool PCASim for cycle-accurate modeling of computing systems with novel architectural features.
- Develop simulation models for the Fresh Breeze system architecture. The simulation models are designed with our own scalable meta-simulator description language, and can be straightforwardly translated into hardware implementation with future refinement.
- Design the funJava programming language which provides data streams as first class data objects, making the language especially suited for DDDAS.
- Develop the Fresh Breeze compiler for generating Fresh Breeze machine codelets from source programs written in funJava, a functional programming language.
- Implement selected test programs in funJava for demonstrating performance and energy efficiency of the Fresh Breeze architecture.

- Develop compilation and program transformation techniques to translate data-flow programs into executable codelets for Fresh Breeze simulators.
- Design an abstraction of the Mahali system for controlled collection of space weather data as a DDDAS example, and express it in the Fresh Breeze programming model.
- Study the modeling of multi phase turbulent fluid flows from experimental measurement and through numerical simulation, and characterize use of experimental data to adapt details of the simulation as a DDDAS.

These steps involve use of new approaches to issues of computer system design, programming and application, and are explained briefly in the following paragraphs.

1 Simulation Using PCASim

The available simulation tools for study of computer architecture are mostly limited to exploring variations and extension of popular commercial processors such as Intel x86 products and MIPS. Adapting these tools for a processor with unusual features and ISA would be challenging and impractical. The alternative to these is simulation software based on system descriptions at the level of logic gates or of combinations of registers and combinational logic blocks (RTL). Use of these tools incurs the substantial effort of expressing a design at such a detailed level, and the substantial penalty of slow simulation speed.

We have developed PCASim, a simulation tool that avoids the drawbacks of both of the above approaches and provides means for cycle accurate modeling of novel system architectures. It is an implementation of the system simulation method proposed by Randall Bryant[3] based on modeling the subject system as *Packet Communication Architecture (PCA)* [4].

The user of PCASim describes the subject system as an interconnection of components in which packet communication is the only means of communication among the components. A component in a PCA system receives packets of data from other components through its input ports, processes the packets, and, following a specified time interval, sends packets via its output ports to other components. It is necessary for the architect of the subject system for simulation to specify a processing duration for each component that accurately reflects the delay of a feasible hardware implementation of the component.

We have used PCASim to perform simulation experiments for Fresh Breeze system configurations with as many as 128 processing cores; it is now a useful and reliable tool for novel system modeling and evaluation. Although PCASim is designed so that creating a multi-host version that will run on standard multicore systems is feasible, we have found that our single-host version provides sufficient simulation speed to be adequate for modeling and evaluating the Fresh Breeze systems we anticipate studying.

2 Research on PCA Simulation

Based on the general ideas of packet communication architecture [3] and distributed discrete event simulation‘[5], Robert Pavel has proposed and constructed a parallel discrete event

simulation engine PICASim, a tool for study and development of novel architectures and program execution models at a reasonable scale prior to devoting resources toward full-scale simulation using a tool such as PCASim. For users of PICASim, Robert has developed LADS: A language for expressing system graphs, including but not limited to PICASim graphs, as well as The LADSpiler: A source-to-source compiler that can optimize and translate systems expressed in LADS to system descriptions needed by PICASim..

He has applied this simulation engine to a version of Fresh-Breeze System One in a way that allows the flexibility to evaluate modifications to the architecture and to provide a framework in which different latencies can be parameterized to determine where design changes may be needed to the architecture model to achieve better performance.

Robert has verified and demonstrated that his simulation tool has the ability to model novel execution and architecture models with a good degree of accuracy. He successfully completed his Ph.D dissertation in May of 2015 [6].

3 Fresh Breeze System Architecture

The Fresh Breeze system architecture supports a unique model for all data objects as trees of fixed-size memory chunks, and provides a built-in scheduler for tasks that execute codelets.

Each data object is represented by a tree of fixed-size *chunks* of memory. Each chunk has a unique identifier, its *handle*, that serves as a globally valid means to locate the chunk within the storage system. Each 128-byte chunk is created and filled with data, but frozen before being shared with concurrent tasks. This *write-once* policy eliminates data consistency issues and simplifies memory management by precluding the creation of cycles in the heap of chunks.

Such a memory model provides a global addressing environment, a virtual one-level store, shared by all user jobs and all cores of a many-core multi-user computing system. It may be extended to cover the entire online storage, replacing the separate means of accessing files and databases in conventional systems.

In a Fresh Breeze system, the unit for managing the processing resource is the single execution of a *codelet*, a block of instructions to be executed without interruption once assigned to a processing core by its task scheduler. A task executing a *master codelet* may spawn one or more *worker tasks* that execute *work codelets* independently. A key innovation is use of a special *sync chunk* to collect results from the worker tasks and spawn a task to execute a specified *continuation codelet* when every worker tasks has contributed its result [1]. Through recursive use of this scheme, a program can generate an arbitrary hierarchy of concurrent tasks corresponding to available parallelism in the computation being performed.

Novel features of the multi-core processing chip include: (1) Cache memories are organized around chunks instead of typical cache lines; (2) Processor registers are tagged to flag those holding handles of chunks; (3) an Auto Buffer provides direct access to chunks without the energy and chip area costs of the tag memory used in a conventional level one cache; and (4) A hardware task scheduler implements fast switching among active tasks at each processing core and a task stealing scheme for load distribution among processing cores.

Our first simulation model (System One) has four components: a Processing Unit, a Memory Unit, an AutoBuffer and the architected Task Scheduler. Our second simulation

model (System Two) includes several copies of SystemOne, with packet switched networks for sending memory commands and receiving memory responses between the processing and memory units.

The instruction set of Fresh Breeze processing units is similar to a RISC load/store design, with adaptations for accessing memory chunks and added instructions for spawning and coordinating execution of tasks. For our first simulation experiments the complete instructions set was not implemented. Recent work has extended support in the compiler and simulation model to all Java data types.

4 The funJava Programming language

The funJava programming language is a version of standard Java, restricted to a functional subset. This simple functional programming language is extended by special classes to support data streams as first class data objects [7]. To the best of our knowledge funJava is the first and only programming language to offer this feature. Other work on supporting data streams in programming languages includes the StreamIt language [8], which is an extension of the C programming language using an enhanced runtime library to support basic operations on stream data. A recent development is OpenStreams [9], an enhancement of StreamIt that offers higher level features and better support for composing stream processing modules. For data streams to be first class objects, a programmer must be able to initialize new data streams and terminate others as an application system requires. The greatest flexibility in stream processing requires automatic memory management, including garbage collection, which StreamIt and openStream do not provide. These properties make funJava an excellent language for expressing DDDAS.

The Fresh Breeze memory model offers a natural representation for data streams. A stream is represented by a chain of memory chunks, each containing stream elements and the handle of the next chunk in the chain. The funJava language includes a special Java class for data streams containing methods for the create, append, first and rest stream operators. These methods will be compiled into machine code instructions for operating on streams as sequences of memory chunks.

These methods are good for expressing such processing steps as filtering, clipping, etc. As we will see below, an essential operation for a typical DDDAS is the merging of streams to produce a stream containing an interleaved sequence of all elements of the input streams. In contrast to the other stream operations, the stream merge operation is nondeterminate – separate runs of a program containing the merge operation may not produce the same result: There are many possible interleavings of two streams, any of which is a valid result.

A notable feature of parallel programs written in funJava is that any program not containing a streamMerge operation is guaranteed to be determinate – all runs of the program with the same input will produce the same result. There can be no data races that could lead to anomalous behavior.

5 Research on Programming Model for Stream Computing

Applications that may be called DDDAS involve both high performance computing and real time interaction with elements of an environment. In addition to linear algebra operations using vectors and matrices of numerical data DDDAS require means for expressing and efficiently executing additional forms of data and operations on them.

- Streams of data arriving continuously for processing by cascades of operations.
- Symbolic commands sent in messages to system components.
- Data storage and retrieval including directories for locating named data objects.

We developed a new programming language/model to support stream processing and the programming of DDDAS applications. Here we will use a simple model DDDAS system to illustrate how representative features of DDDAS are expected to be specified/expressed in the programming language. For the Fresh Breeze project the language will be a restricted version of Java, extended with features for streams and transactions such as those used in the examples that follow.

In our simple DDDAS system, a number of sensor components collect data items of interest at locations in the system environment. A stream of commands from a central control element directs each sensor. Each sensor sends, following commands, substantial sequences of similar data items to a processor element. One would like to have the data processed expeditiously, so the processor has an entry component (like the waiting area of a barber shop) where jobs await available resources for the desired processing. Processed data is held on a database and made accessible to research staff for reviewing and evaluation.

From a high-level point-of-view, given the Fresh Breeze memory model in which data objects are represented by trees of fixed-size chunks, the natural choice for streams is a linear chain of chunks. Each chunk will hold several data items and a reference to the next chunk in the chain. Stream elements are removed from the head chunk of the chain, and new elements are added to the tail chunk, adding a new chunk to the chain if the tail chunk is full. Moreover, we provide the synchronization mechanism in stream processing with a special object known as a future. The four operations on streams: new, first, rest, and append, are supported by instructions of the Fresh Breeze instruction set.

Figure 1 illustrates our programming model for a kind of requirement that is likely to be present in many DDDAS. The computation pattern includes the selection of a stream of data for processing from multiple streams of data items.

6 The Fresh Breeze Compiler

The Fresh Breeze Compiler uses the standard Java compiler (javac) to produce class files that are processed in three phases:

```
class TransactionManager <RQ> {

    private Guard managerGuard;
    private Stream <QueueEntry> queueHead;
    private Stream <QueueEntry> queueTail;

    // A class for the queue of requests
    class QueueEntry {
        RQ request;
        Future <QueueEntry> next;

        QueueEntry ( RQ request ) {
            this.request = request;
            next = new Future <QueueEntry> ();
        }
    }

    // Constructor
    TransactionManager () {
        queueTail = new Stream <QueueEntry> ();
        queueHead = queueTail;
        Guard managerGuard = new Guard (queueTail);

        processQueue (queueHead);
    }

    public void enterRequest ( RQ request ) {
        RequestEntry newEntry = new QueueEntry ( request );
        oldTail = managerGuard.guardSwap (newEntry.next);
        FutureWrite (oldTail, newEntry);
    }

    private void processQueue (Stream <QueueEntry> reqQueue) {
        QueueEntry entry = reqQueue.futureRead ();
        <RQ> request = entry.request;
        request.processRequest();
        Future <QueueEntry> newHead = entry.next;
        processQueue (newHead);
    }

} // end class TransactionManager
```

Figure 1: A Programming Model example—the transaction manager. The two operations guardSwap and futureWrite in the enterRequest method perform a lock-free insert of a new request in the request queue.

- **Convert** This component converts the Java bytecode of each method in a class file into a data flow graph that represents the method. The elements of the data flow graphs are unary and binary operators, array get and set operators, conditionals, and while or until loops.
- **Transform** The data flow graph of each method is analyzed to identify opportunities for exploiting parallelism. In particular, loops that perform data parallel reductions or array construction are recognized. Then a set of data flow graphs is constructed that

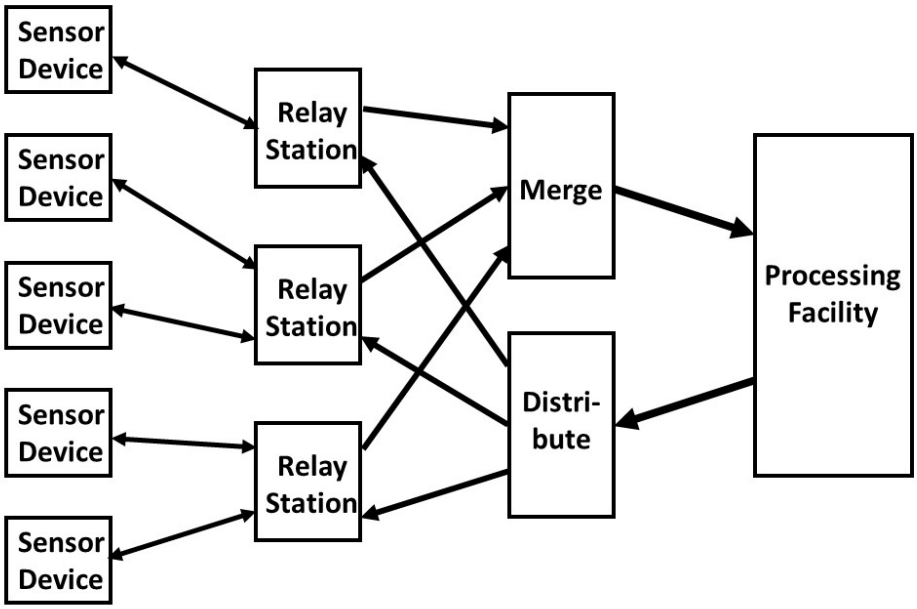

Figure 2: Abstraction of the Mahali vision as DDDAS.

represents the several codelets needed to implement the parallelism of the method.

- **Construct** Each data flow graph representing a codelet is converted into a block of machine instructions that defines a codelet ready for execution.

The Fresh Breeze compiler has two unusual features. First, it uses data flow graphs as its internal representation for programs, a choice that had been adopted earlier in compilers for the Sisal programming language![10]. Secondly, because code is generated separately for each codelet, and the size of each codelets is relatively small, a detailed analysis of each codelet is performed, then optimized machine code is constructed, as reported in [11]. Thus our development of the FrBzCompiler represents a significant investment in compiler functionality that we expect could well be a model for future compilers for systems that adopt a codelet model for parallel computation.

7 Mahali Controlled Data Collection

The Mahali Project at the MIT Haystack Observatory [12] is developing a plan for monitoring *space weather*, the fluctuation of electron density in Earth's ionosphere as a result of solar events, and also from changes at the Earth's surface. Data is gathered from widely dispersed sensors and processed by central computing facilities for presentation to and interpretation by scientists. Data on electron density is obtained by measuring properties of the navigation signals transmitted by GPS satellites, which are affected by their transmission through the ionosphere.

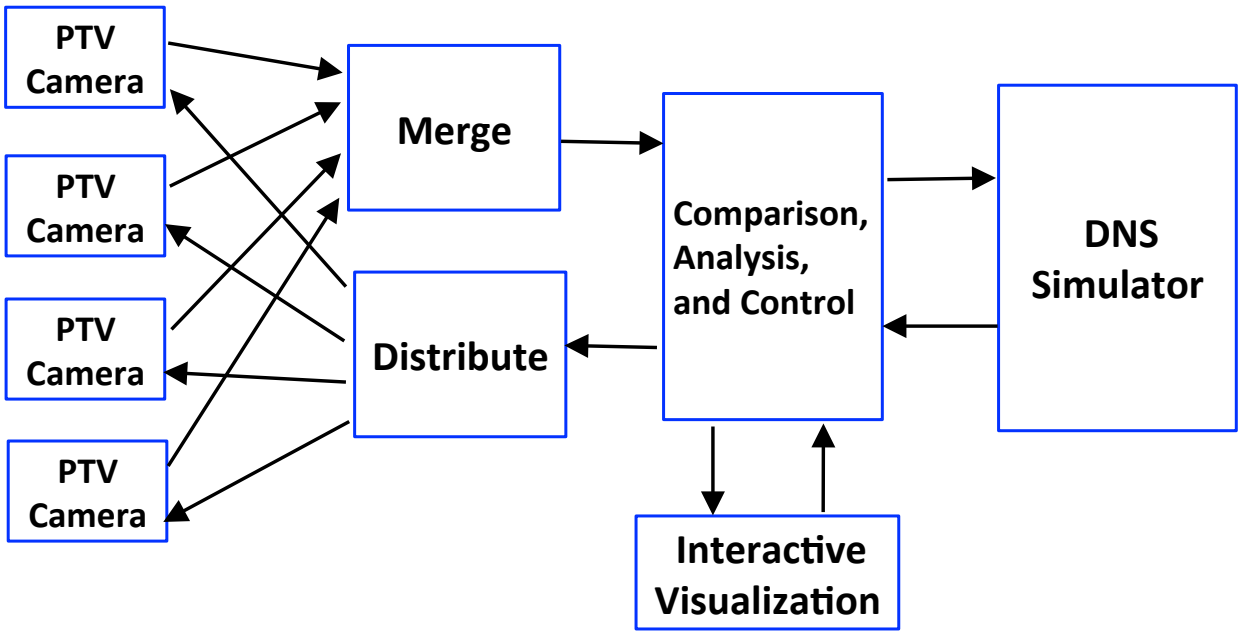

Figure 3: Simulation and experiment for multi-phase flow as a DDDAS.

Because the sensors are deployed in remote locations they are powered by solar cells. In consequence, the GPS receivers must be turned on only when interesting data will be collected and sufficient energy is available to operate them. To provide for controlling data collection and making inquiries about the status of sensors, communication of commands from the data processing center to the sensors is needed. It has been noted that the ubiquity of mobile communication devices offers the possibility to utilize these devices to relay data and commands between sensors and processing centers, thereby avoiding the expense of setting up and operating a specialized communication network. With these characteristics, the Mahali network and computation nodes form a fine example of DDDAS.

We have chosen to use the controlled data collection of the Mahali vision as inspiration for a model DDDAS to demonstrate the benefits of the Fresh Breeze programming model and system architecture for DDDAS. Figure 2 shows the Mahali abstraction we are exploring. Many Sensor Devices send data to and receive commands from a Processing Facility by way of mobile Relay Stations. The activity of each component of this model other than the Merge module can be expressed as a computation on input streams to produce an output stream. The stream merge operation is used to combine streams of data for transmission by a relay station and for processing by the central processing facility. The abstraction of Figure 2 incorporates all essential features of Mahli data collection and is a natural fit to the Fresh Breeze programming model.

8 Modeling of multiphase flows using a DDDAS-based systems approach

Turbulent multiphase flows occur in gas turbines used for power generation, aircraft engines operating in a polluted environment, as a result of explosions, either deliberate or accidental, and in other industrial operations. This work concerns the efficient and accurate modeling of these multiphase flows to guide the design of safer and more effective equipment for military and industrial uses.

The UDel team (Wang in Computational Multiphase Flow Lab in Mechanical Engineering and Yu in Computer Graphics Lab in Computer and Information Sciences) are developing two advanced approaches for studying flows of particle-laden fluids. On one hand, numerical simulation techniques and the advent of increasingly powerful computer systems have made fluid simulation at the mesoscopic level practical. The lattice Boltzmann method is now a feasible approach for these problems. The other approach is experimental and uses a group of fast cameras to collect data on particle movements and fluid velocity in a test flow field. This technique is known as Plenoptic Particle Tracking Velocimetry (PPTV).

Work in Wang's group at UDel has implemented many improvements to the lattice Boltzmann method (LBM), and has also developed a new mesoscopic simulation method known as the Discrete Unified Gas Kinetic Scheme (DUGKS). Specific research activities include:

1. Using the LBM code, we studied different profiles conditioned on the particle surface for forced particle-laden turbulence. The results have been published in ASME J. Fluids Engineering.

2. We investigated a number of implementation issues related to the lattice Boltzmann simulation of moving particles in a viscous flow: (a) creation of missing populations when a new fluid node is created; (b) the calculation of force and torque acting on the moving particles, and (c) numerical instability. We also explored local grid refinement to reduce force fluctuations. These efforts along with several benchmark case-study results have been reported in two accepted journal papers.

3. The major task in the past year was to publish results from particle-resolved simulation of particle-laden turbulent channel flow using the mesoscopic lattice Boltzmann method. Two journal papers have been published along this line, reporting results related to turbulence modulation by finite-size particles, particle concentration distribution, and particle slip velocity on the channel wall.

4. Another major effort was to optimize the LBM particle-resolved code. A sophomore undergraduate researcher examined each subroutine for the particulate phase, and was able to reduce the computational overhead for the particulate phase from 50% to 20%. In addition, he explored several ways of combining collision and streaming substeps to speed up the flow-simulation part of the code. Overall, our simulation code is now four-time more efficient.

5. Another exciting accomplishment is the development of a second mesoscopic approach - the Discrete Unified Gas Kinetic Scheme (DUGKS) for turbulent flows. Unlike LBM,

DUGKS is based on a finite-volume formulation of the Boltzmann equation, which has two significant advantages: DUGKS can handle compressible turbulence and even non-continuum flows, DUGKS allows the use of a non-uniform grid structure. This can greatly expand the application domains of our mesoscopic DNS tools. So far, we have developed scalable DUGKS codes for both homogeneous flows and for single-phase turbulent channel flow.

6. We studied the appropriate parallelization schemes for the LBM code. Particularly, we explored the use of GPU (Graphic Processing Units) to accelerate several most computational intensive kernels in the LBM code. We also experimented generalizing the parallel computation pattern in a new codelet based parallel execution model—Fresh Breeze. Furthermore, we started the research on the interactive visualization of high-performance simulation code.

7. We have recently built a closed-loop water channel which will soon be used to validate the accuracy of a PPTV measurement system. The first step will be to compare measurements of single-phase turbulent open channel flow with data from LBM simulations. The second step will be to introduce solid particles into the turbulent flow and again compare measurement data with LBM simulations.

Both numerical simulation and PTV have limitations; however, their use together provides a new opportunity for rigorous study of turbulent multiphase flows. Numerical simulations may be used to calibrate experimental measurements using PTV; conversely, the PTV data can show where simulations are inaccurate, for example, at boundary layers of fluid flow.

Our future work will involve this joint application of experimental and numerical modeling techniques as a dynamic data driven application system, as shown in Figure 3. An Analyzer/Controller receives data streams from PTV cameras, monitors a concurrent numerical simulation, and looks for discrepancies between the experimental data and computed results. The PTV cameras are sent new commands and simulation parameters are adjusted accordingly. Results are presented to the visualization station for operators to review and respond.

9 Publications

1. Daniel A. Orozco, Elkin Garcia, Robert S. Pavel, Jaime Arteaga, Guang R. Gao: The Design and Implementation of TIDeFlow: A Dataflow-Inspired Execution Model for Parallel Loops and Task Pipelining. International Journal of Parallel Programming 44(2): 278-307 (2016)

2. Jack B. Dennis. Compiling Fresh Breeze Codelets PMAM'14 Proceedings of Programming Models and Applications on Multicores and Manycores. ACM New York, NY. 2014.

3. Jack B. Dennis. A Fresh Breeze Processor Supporting Instruction Level Parallelism. Workshop on Data-Flow Execution Models for Extreme Scale Computing (DFM 2013).

4. Jack B. Dennis and Guang R. Gao. Design for a Multi-Core OS. Forth Workshop on Data-Flow Execution Models for Extreme Scale Computing (DFM 2014), August 24, 2014, Edmonton, Alberta, Canada.

5. Jack Dennis. Fresh Breeze. Execution Models and Runtime Systems, 2014 ACS Productivity Workshop. 5520 Research Park Drive, Catonsville, MD July 15-17, 2014

6. Guang R. Gao and Jack B. Dennis. Massively Multi-Core Systems and Virtual Memory CAPSL Technical Memo 128 April 29th, 2014

7. Jack B. Dennis. IFIP Working Group 2.8 Functional Programming. October 14 - 18, 2013 Meeting, Assois, France August 10 - 15, 2014 Meeting, Estes Park, Colorado May 25 - 29, 2015, Kefalonia, Greece

8. Chen SY, Peng C, Teng YH, Wang L-P, 2015, Improving lattice Boltzmann simulation of moving particles in a viscous flow using local grid refinement, Computers and Fluids, submitted.

9. Lin, Zhaowu; Shao, Xueming; Yu, Zhaosheng; Wang, L-P, 2015, Effects of finite-size neutrally buoyant particles on the turbulent flows in a square duct. Submitted to J. Fluid Mech., JFM-15-S-0787.

10. Peng C, Teng Y, Hwang B, Guo ZL, Wang L-P, 2015, Implementation issues and benchmarking of lattice Boltzmann method for moving particle simulations in a viscous flow, Computers and Mathematics with Application, in press. doi:10.1016/j.compfluid.2015.08.027

11. Wang L-P, Peng C, Guo ZL, Yu ZS, 2015, Lattice Boltzmann Simulation of Particle-Laden Turbulent Channel Flow, Computers and Fluids, in press. doi:10.1016/j.compfluid.2015.07.008

12. Wang L-P, Ardila OGC, Ayala O, Gao H, Peng C, 2015, Study of local turbulence profiles relative to the particle surface in particle-laden turbulent flows, Submitted to ASME J. of Fluids Engr., in press. doi: 10.1115/1.4031692.

13. Wang L-P, Peng C, Guo ZL, Yu ZS, 2015, Flow Modulation by Finite-Size Neutrally Buoyant Particles in a Turbulent Channel Flow, Submitted to ASME J. of Fluids Engr., in press. doi: 10.1115/1.4031691.

14. Yu ZS, Lin ZW, Shao XM, Wang L-P, A parallel fictitious domain method for the interface-resolved simulation of particle-laden flows and its application to the turbulent channel flow, Engr. Appl. Comput. Fluid Mech., Accepted.

15. Zong Y., Guo ZL, Wang L-P, 2015, Designing Correct Fluid Hydrodynamics on A Rectangular Grid using MRT Lattice Boltzmann Approach, Computers and Mathematics with Application, in press. doi:10.1016/j.camwa.2015.05.021

10 Conference Presentations

1. Xiaoming Li. FreshBreeze: A Data Flow Approach for Meeting DDDAS Challenges. Xiaoming Li, Jack Dennis, Guang Gao, Willie Lim, Haitao Wei, Chao Yang, Robert Pavel. DDDAS-Dynamic Data Driven Applications Systems and Large-Scale-Big-Data & Large-Scale-Big-Computing (DDDAS-LS). ICCS 2015, June 2015. Reykjavk, Iceland.

2. Bo YT, Wang P, Guo ZL, Wang L-P, Parallel implementation and validation of DUGKS for three-dimensional Taylor-Green vortex flow and turbulent channel flow. Presented at The Twelfth International Conference for Mesoscopic Methods in Engineering and Science, Beijing, China, July 20-24, 2015. Submitted to Computers and Fluids.

3. Min H, Peng C, Wang L-P, Guo ZL, An inverse design analysis of forcing terms in MRT lattice Boltzmann models. Presented at The Twelfth International Conference for Mesoscopic Methods in Engineering and Science, Beijing, China, July 20-24, 2015. Submitted to Computers and Fluids.

4. Peng C, Guo ZL, Wang LP, A lattice-BGK model for the Navier-Stokes equation based on a rectangular grid. Presented at The Twelfth International Conference for Mesoscopic Methods in Engineering and Science, Beijing, China, July 20-24, 2015. Submitted to Computers and Fluids.

5. Peng C, Wang L-P, Guo ZL, Yu ZS, 2015, two-way interactions in particle-laden turbulent channel flow. AJK2015-32672, Proceedings of the ASME-JSME-KSME Joint Fluids Engineering Conference 2015, July 26-31, 2015, Seoul, Korea.

6. Wang L-P, Min H, Peng C, A lattice-Boltzmann scheme of the Navier-Stokes equation on a 3D cuboid lattice. Presented at The Twelfth International Conference for Mesoscopic Methods in Engineering and Science, Beijing, China, July 20-24, 2015. Submitted to Computers and Fluids.

7. Wang P, Guo, ZL; Xu K; Wang LP, A comparative study of DUGKS and LBE methods for laminar flows and decaying homogeneous isotropic turbulence flows. Presented at The Twelfth International Conference for Mesoscopic Methods in Engineering and Science, Beijing, China, July 20-24, 2015. Submitted to Computers and Fluids.

8. Wang L-P, Peng C, and Guo ZL, Study of turbulence modulation by finite-size solid particles using the Lattice Boltzmann approach. AJK2015-32671, Proceedings of the ASME-JSME-KSME Joint Fluids Engineering Conference 2015, July 26-31, 2015, Seoul, Korea.

11 Project Personnel and Contributions

Xiaoming Li, contributed to the designing and developing compilation and code transformation techniques based on data-flow graph representation of programs. Moreover, Li worked

with Wang on various DDDAS application-specific optimization and data-driven interaction simulation techniques. He also coordinated the project and organized project meetings.

Guang Gao, contributed to the simulation of the Fresh Breeze architecture, the data-flow based stream processing programming model, and the evaluation of the initial system implementation. Gao also was in charge of the development of Fresh Breeze architecture backend on new many-core computers. In addition, he contributed to the development of application-specific optimization techniques.

Jack Dennis, was in charge of the Fresh Breeze architecture design, the architecture simulation and the architect of the compiler. In addition, Dennis proposed the new programming model in the context of Fresh Breeze. He also implemented the compilers frontend and backend as well as various other components.

Lian-Ping Wang, Wang was in charge of developing a particle-resolved simulation for particle-laden turbulent flow using the lattice Boltzmann method, participated in discussions of data-driven interactive simulations, and helped test visualization tools. He also took charge of organizing monthly group meetings.

Haitao Wei, Postdoc researcher, supported on the grant. Wei helped coordinating the development of codelet transformation techniques and contributed to the implementation of various compiler components.

Tom St. John, Ph.D. student, supported on the grant. He contributed to the performance evaluation of the initial implementation of Fresh Breeze simulation.

Chao Yang, Ph.D. student, supported on the grant. Yang developed the interfaces between the codelet transformer with upstream and downstream compiler components. He also ported the transformer between different generations of simulation, and helped testing various compiler components.

Cheng Peng, Ph.D. student, supported on the grant. Peng worked with Wang to code and debug the lattice Boltzmann method, solved problems related to the code, and presented results at the group and the ICMMES2014 meeting.

Yuan Zong, Visiting Professor from China, self-supported. She is developing and testing a new lattice Boltzmann scheme that could be used with a rectangular grid structure. Presented a paper at the ICMMES2014 meeting.

Songying Chen, Visiting Professor from China, self-supported. He is developing a local grid refinement method to improve the accuracy of the simulations. He also presented a paper at the ICMMES2014 meeting.

Yihua Teng, self-supported, a senior visitor from Peking U. China. He helped develop a few benchmark cases for lattice Boltzmann simulations.

References

[1] J. B. Dennis, "The Fresh Breeze model of thread execution," in *Workshop on Programming Models for Ubiquitous Parallelism*, IEEE, 2006. Published with PACT-2006.

[2] J. B. Dennis, "A parallel program execution model supporting modular software construction," in *Massively Parallel Programming Models*, pp. 50–60, IEEE, 1997.

[3] R. E. Bryant, "Simulation of packet communication architecture computer systems," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1977.

[4] J. B. Dennis, "Packet communication architecture," in *Proceedings of the 1975 Sagamore Computer Conference on Parallel Processing*, pp. 224–229, IEEE, 1975.

[5] K. Chandy and J. Misra, "Distributed simulation: A case study in design and verification of distributed programs," *Software Engineering, IEEE Transactions on*, vol. SE-5, pp. 440 – 452, sept. 1979.

[6] R. Pavel, *Simulation Methodology and Tools for the Development of Novel Program Execution Models and Architectures*. PhD thesis, University of Delaware, May 2015.

[7] J. B. Dennis, *Stream data types for signal processing*. IEEE Computer Society Press, 1995.

[8] W. Thies, M. Karczmarek, and S. P. Amarasinghe, "Streamit: A language for streaming applications," in *Proceedings of the 11th International Conference on Compiler Construction*, (London, UK, UK), pp. 179–196, Springer-Verlag, 2002.

[9] A. Pop and A. Cohen, "Openstream: Expressiveness and data-flow compilation of openmp streaming programs," *ACM Trans. Archit. Code Optim.*, vol. 9, pp. 53:1–53:25, Jan. 2013.

[10] J. McGraw, S. Skedzielewski, S. Allan, R. Oldehoeft, J. Glauert, C. Kirkham, B. Noyce, and R. Thomas, "Sisal: Streams and iteration in a single assignment language," Technical Report M-146, Rev. 1., Lawrence Livermore National Laboratory, Livermore, CA, 1985.

[11] J. B. Dennis, "Compiling Fresh Breeze codelets," in *Proceedings of Programming Models and Applications on Multicores and Manycores*, PMAM'14, (New York, NY, USA), pp. 51:51–51:60, ACM, 2014.

[12] V. Pankratius, F. Lind, A. Coster, P. Erickson, and J. Semeter, "Mobile crowd sensing in space weather monitoring: The Mahali project," *Communications Magazine*, vol. 52, pp. 111–133, Aug 2014.

1.

1. Report Type

Final Report

Primary Contact E-mail
Contact email if there is a problem with the report.

xli@udel.edu

Primary Contact Phone Number
Contact phone number if there is a problem with the report

302-831-0334

Organization / Institution name

University of Delaware

Grant/Contract Title
The full title of the funded effort.

System Architecture for High Performance Interactive Computation

Grant/Contract Number
AFOSR assigned control number. It must begin with "FA9550" or "F49620" or "FA2386".

FA9550-13-1-0213

Principal Investigator Name
The full name of the principal investigator on the grant or contract.

Xiaoming Li

Program Manager
The AFOSR Program Manager currently assigned to the award

Dr. Frederica Darema

Reporting Period Start Date

09/30/2015

Reporting Period End Date

03/29/2016

Abstract

This is the final report of our project. In this report, we will cover our research activities in the last report period, as well as summarize our achievements for the whole project.

Our research effort has been directed at developing an efficient system architecture and software tools for building and running Dynamic Data Driven Application Systems (DDDAS). The foremost requirement for efficient operation of DDDAS is flexibility and efficiency of managing memory and processing resources for an operating environment char- acterized by continuously changing resource demands. The Fresh Breeze programming model [1][2] is well matched to this application environment.

With generous support from AFOSR, and additional support from NSF, we have ac- complished the following steps toward an energy efficient computing platform that directly implements a programming model well suited to DDDAS.

• Create the simulation tool PCASim for cycle-accurate modeling of computing systems with novel architectural features.
• Develop simulation models for the Fresh Breeze system architecture. The simulation models are

designed with our own scalable meta-simulator description language, and can be straightforwardly translated into hardware implementation with future refinement.
• Design the funJava programming language which provides data streams as first class data objects, making the language especially suited for DDDAS.
• Develop the Fresh Breeze compiler for generating Fresh Breeze machine codelets from source programs written in funJava, a functional programming language.
• Implement selected test programs in funJava for demonstrating performance and energy efficiency of the Fresh Breeze architecture.
• Develop compilation and program transformation techniques to translate data-flow programs into executable codelets for Fresh Breeze simulators.
• Design an abstraction of the Mahali system for controlled collection of space weather data as a DDDAS example, and express it in the Fresh Breeze programming model.
• Study the modeling of multi phase turbulent fluid flows from experimental measurement and through numerical simulation, and characterize use of experimental data to adapt details of the simulation as a DDDAS.

Distribution Statement

This is block 12 on the SF298 form.

Distribution A - Approved for Public Release

Explanation for Distribution Statement

If this is not approved for public release, please provide a short explanation. E.g., contains proprietary information.

SF298 Form

Please attach your SF298 form. A blank SF298 can be found here. Please do not password protect or secure the PDF The maximum file size for an SF298 is 50MB.

AFD-070820-035.pdf

Upload the Report Document. File must be a PDF. Please do not password protect or secure the PDF . The maximum file size for the Report Document is 50MB.

completed-work.pdf

Upload a Report Document, if any. The maximum file size for the Report Document is 50MB.

Archival Publications (published) during reporting period:

2. New discoveries, inventions, or patent disclosures:
Do you have any discoveries, inventions, or patent disclosures to report for this period?

No

Please describe and include any notable dates

Do you plan to pursue a claim for personal or organizational intellectual property?

Changes in research objectives (if any):

Change in AFOSR Program Manager, if any:

Extensions granted or milestones slipped, if any:

The project is granted a 6-month no-cost-extension from 09/30/15-03/29/16.

AFOSR LRIR Number

LRIR Title

Reporting Period

Laboratory Task Manager

Program Officer

Research Objectives

Technical Summary

Funding Summary by Cost Category (by FY, $K)

	Starting FY	FY+1	FY+2
Salary			
Equipment/Facilities			
Supplies			
Total			

Report Document

Report Document - Text Analysis

Report Document - Text Analysis

Appendix Documents

2. Thank You

E-mail user

Jun 27, 2016 21:11:45 Success: Email Sent to: xli@udel.edu